APPLICATION PROPAGATION ENVIRONMENT (APEX)

Interoperability and Compliance Guidelines

vito

Vision on technology
for a better world

vito.be

DOCUMENT CONTROL

Document Title	Interoperability and Compliance Guidelines
Project	Application Propagation Environment (APEx)
Version	3.1

DOCUMENT APPROVER(S) AND REVIEWER(S)

Version	Name	Company	Action	Date
2.1	Patrick Griffiths Paulo Sacramento	ESA	Approved	26/08/2024
2.4	Patrick Griffiths Paulo Sacramento	ESA	Approved	12/12/2024
3.1	Patrick Griffiths Paulo Sacramento	ESA	Approved	10/07/2025

DOCUMENT VERSION HISTORY

Version	Date	Section	Description	Editor/Reviewer
1.0	18/03/2024	All	Initial content	Bram Janssen
1.0	28/03/2024	All	Updated content based on comments	Jeroen Dries
1.0	12/04/2024	2	Updated broken reference	Bram Janssen
2.0	27/06/2024	2,3	Added content based on GitHub repository	Bram Janssen
2.0	03/07/2024	4	Added content for Dashboards	Adam Tweedie
2.1	05/08/2024- 14/08/2024	All	Updated content based on MS1 RIDs V1.0	Bram Janssen, Jeroen Dries, Pedro Gonçalves
2.2	24/10/2024	All	Updated content based on latest version on APEx documentation	Bram Janssen
2.3	12/11/2024	All	Updated content based on comments SRR	Bram Janssen
2.4	04/12/2024	All	Updated content based on feedback of ESA	Bram Janssen Jeroen Dries Pedro Gonçalves Hervé Caumont
3.0	24/05/2025	5.5	Added section on supported platforms	Bram Janssen

3.0	08/05/2025	6	Updated requirements for Geospatial Explorer	Dan Ormsby Adam Tweedie
3.0	09/05/2025	4,5	Synchronized with online content	Jeroen Dries
3.1	26/05/2025	Preamble, 4, 5	Updated content based on MTR RIDs V1.0	Bram Janssen Jeroen Dries

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

NORMATIVE REFERENCES

The documents referenced in the following chapters are cited in such a way that some or all of their content supports the requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document, including any amendments, applies.

[1] “OpenEO User Defined Processes,” [Online]. Available: https://open-eo.github.io/openeo-python-client/udp.html.

[2] “OGC Best Practice for Earth Observation Application Package,” [Online]. Available: https://docs.ogc.org/bp/20-089r1.html.

[3] “Commonwl.org: Common Workflow Language Specifications,” [Online]. Available: https://w3id.org/cwl/.

[4] “OGC API — Processes — Part 1: Core Standard, 2021,” [Online]. Available: https://docs.ogc.org/is/18-062r2/18-062r2.html.

[5] “openEO,” [Online]. Available: https://openeo.org/.

[6] “OGC Cloud Optimized GeoTIFF,” [Online]. Available: https://docs.ogc.org/is/21-026/21-026.html.

[7] “OGC CF-netCDF,” [Online]. Available: https://www.ogc.org/standard/netcdf/.

[8] “Geozarr Specification,” [Online]. Available: https://github.com/zarr-developers/geozarr-spec.

[9] “GeoJSON,” [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7946.

[10] “GeoParquet 1.1.0,” [Online]. Available: https://geoparquet.org/releases/v1.1.0/.

[11] “STAC Specification,” [Online]. Available: https://github.com/radiantearth/stac-spec/.

[12] “OpenEO API 1.2.0,” [Online]. Available: https://openeo.org/documentation/1.0/developers/api/reference.html.

[13] “OpenEO API Profiles,” [Online]. Available: https://openeo.org/documentation/1.0/developers/profiles/api.html.

[14] “OpenEO processes profiles,” [Online]. Available: https://openeo.org/documentation/1.0/developers/profiles/processes.html.

[15] “WMS 1.3.0,” [Online]. Available: https://portal.ogc.org/files/?artifact_id=14416.

[16] “WMTS 1.0.0,” [Online]. Available: https://portal.ogc.org/files/?artifact_id=35326.

[17] “WFS 2.0,” [Online]. Available: https://docs.ogc.org/is/09-025r2/09-025r2.html.

[18] “OGC API Features 1.0,” [Online]. Available: https://www.ogc.org/publications/standard/ogcapi-features/.

[19] “Radiant Earth Foundation: SpatioTemporal Asset Catalog specification,” [Online]. Available: https://stacspec.org.

[20] “STAC projection extension,” [Online]. Available: https://github.com/stac-extensions/projection.

[21] ”PROJ4JS” [Online]. Available: http://proj4js.org/

1 INTRODUCTION

The following text provides an overview of the APEx interoperability and compliance guidelines. These guidelines are specifically aimed at both algorithm providers, data providers and algorithm hosting platform providers.

This document starts by explaining the general concepts and terms that will be used throughout the text. Understanding these concepts is essential for comprehending the subsequent chapters' guidelines and gaining a clearer understanding of APEx's roles and responsibilities. The document also includes a list of online references supporting the outlined requirements.

The guidelines then outline specific requirements for algorithm providers. This includes guidelines on how to structure, implement, and document algorithms. These guidelines were designed with the explicit goal of facilitating the streamlined integration of algorithms and workflows into APEx.

Furthermore, this text sets out essential requirements for algorithm hosting platforms. Ensuring compliance with these requirements simplifies their integration with APEx. Consequently, this ensures hosting platforms can contribute more effectively to APEx more efficiently and successfully.

The guidelines in this document also define combined service level commitments (SLA and OLA), including accompanying key performance indicators (KPIs) and metrics. These will serve to validate and ensure the compliance of these commitments.

Lastly, this guide explains in detail the federated business model utilised within APEx. Through providing comprehensive documentation of this model, APEx aims to foster a precise understanding among all stakeholders and create a cooperative environment.

2 DEFINITIONS & ACTORS

This section introduces key terms and their meaning in the APEx context, which are crucial for building a common understanding of the complex ecosystem in which APEx operates. Along with the general definitions, Figure 1 visually demonstrates the interconnections between these concepts.

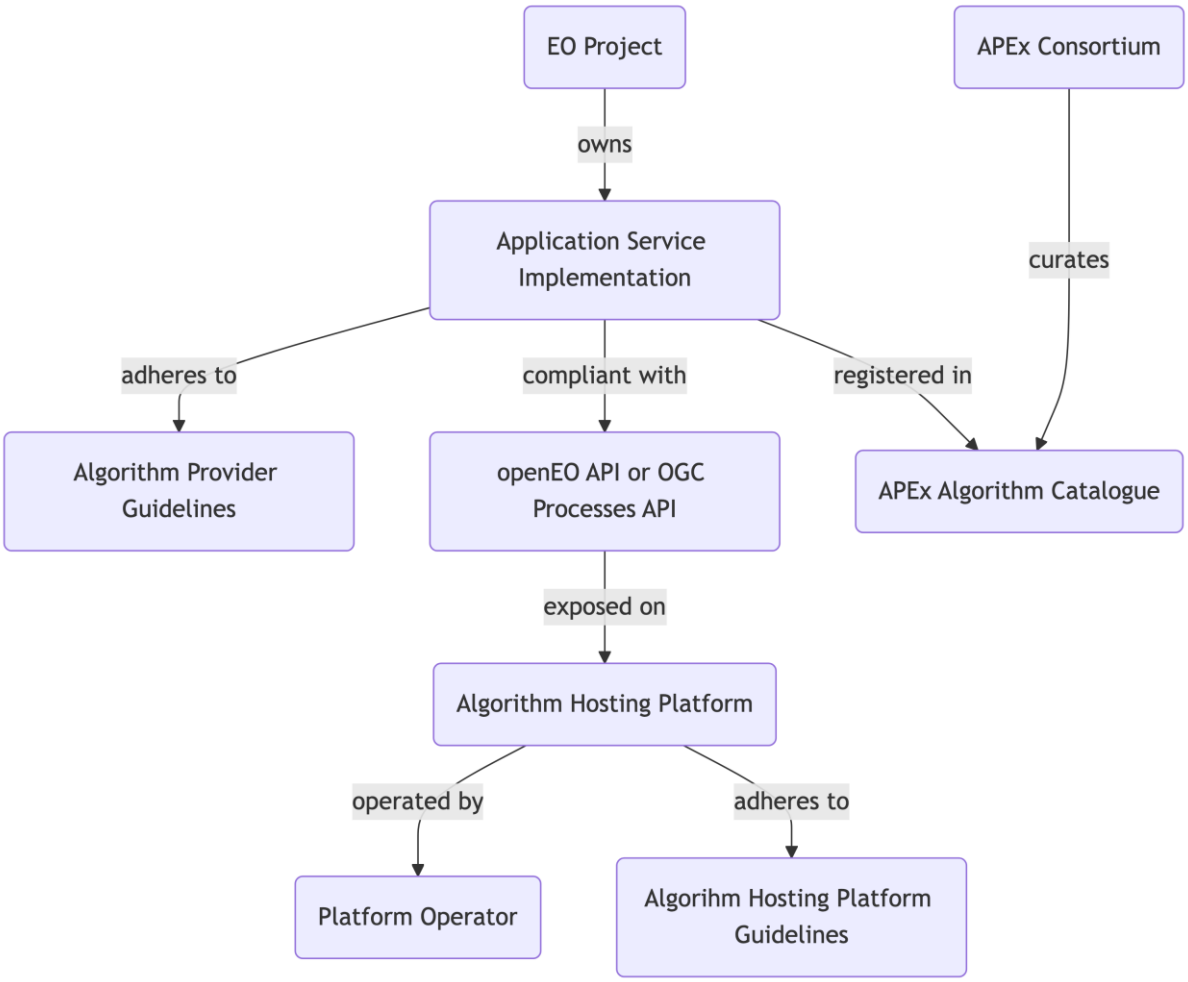

Figure 1. Concepts and relationships

2.1 Concepts

2.1.1 EO Algorithms, Workflows and Applications

In the context of APEx, the term algorithm refers to a specific piece of EO software that performs specific processing or data analytics tasks. EO Algorithms can vary substantially in size and complexity, for example, some only perform one specific task (e.g. cloud masking in optical imagery), while others generate value-added products based on complex workflows that integrate several individual algorithms. A related term is EO Application, which commonly refers to a comprehensive workflow implementation that commonly also utilises non-EO (geospatial) data sets and a variety of data analytics procedures to perform an application-specific processing task.

For the sake of simplicity in the context of APEx, we refer to these simply as (EO) “algorithms”.

2.1.2 Algorithm Service Implementation

The algorithm definition refers to a representation of the algorithm modules and interfaces that can be exposed by an API and/or processing platform. Typically, it includes a general description of the algorithm along with detailed information on its parameters, expected output, scientific method, and an overview of the different steps executed within the algorithm. Examples of algorithm definitions include openEO’s User Defined Processes (UDP) [1] and OGC Application Package [2], using the Common Workflow Language (CWL) [3].

2.1.3 Algorithm Catalogue

The APEx algorithm catalogue is a central register of algorithm definitions and the corresponding algorithm service instances that can be executed on APEx-compliant algorithm hosting platforms. Curated by APEx, the catalogue relies on automated checks to ensure that advertised algorithms service instances are available and functional. Whenever a malfunction is detected, this is reported to ESA and the EO project consortium, allowing them to determine a proper course of action.

2.1.4 Hosted Algorithm

To increase uptake and interoperability, APEx aims to enable the execution of algorithms via standardised web service APIs. This transitions algorithms from being rather arbitrary pieces of software with potentially complex requirements, in terms of execution environment, usage, inputs, …, into on-demand services that can readily be invoked by stakeholders. This transformation primarily involves converting a given algorithm into an APEx-compliant algorithm definition and making it available as a service on an algorithm hosting platform. The transition process into a hosted on-demand algorithm service is supported by the APEx Algorithm Services.

An important boundary condition for hosted algorithms is that they can be executed at a predictable cost for a given set of inputs. This predictability allows a service user to accurately estimate and determine the cost associated with the execution of the final deployed service instance.

2.1.5 Algorithm Hosting Platform

An EO algorithm hosting platform enables the execution of a standardised algorithm, represented by an algorithm definition. In APEx, an algorithm hosting platform specifically refers to platforms that support the openEO UDP and/or OGC Application Package Algorithm description standards. These platforms also enable algorithms to be executed by the openEO API or through OGC API Processes. For APEx, these platforms are considered existing providers available through ESA’s Network of Resources (NoR). Examples of such algorithm hosting platforms are the Copernicus Data Space Ecosystem for openEO or Ellip for OGC Application Packages.

It is important to note that APEx itself is not an algorithm hosting platform; rather, it promotes the reuse of existing platforms. A key property of algorithm hosting platforms is their long-term sustainability beyond the lifetime of a typical EO project. This ensures that algorithms can still be executed after the project ends.

The algorithm hosting platform has an important responsibility to ensure the continued availability of hosted algorithms. This responsibility is detailed in the requirements below, highlighting that the selection of the platform affects properties such as cost, performance, stability, and the amount of computing resources available to run the algorithm. Compliance with these requirements does not necessarily imply a high overall quality level across all aspects, ensuring that EO projects retain a sufficient degree of freedom in selecting their preferred platform.

2.2 Actors

2.2.1 EO Project & Algorithm PI

An EO project refers to the consortium that is responsible for building the EO algorithm and is also referred to as Algorithm PI (principal investigator). In certain ESA EOP procurement and ITTs, there is now a requirement included to comply with APEx interoperability and compliance guidelines. When compliance is required, the consortium can utilise various services offered by APEx. Specifically, the APEx Algorithm Services aim to support the enhancement of algorithms on a technical and software level and facilitate the transition to hosted algorithms that can be included in the APEx algorithm catalogue.

ESA EO projects that do not have an explicit compliance requirement are also eligible to receive support. The APEx support can boost project impact, so projects are encouraged to inquire with their ESA technical officer about the possibilities.

It is important to note that during the execution of the project, the project retains full responsibility for the final quality of the algorithms and workflows.

2.2.2 APEx Consortium

The APEx team, comprised of industry experts, operates the various services provided by APEx. To maximise the reuse of existing resources, the team leverages service offerings within ESA's Network of Resources (NoR), drawing on the extensive ecosystem provided by the EO industry.

Although members of the APEx consortium are involved in various EO services registered in NoR, APEx itself is not a new EO platform. Instead, it focuses on enabling interoperability among existing platforms. As a result, APEx remains open to integrating additional platforms, provided they meet the compliance requirements of the Algorithm Hosting Environment, as specified below.

2.2.3 Platform Operator

The platform operator plays a crucial role in managing and running the algorithm hosting platform. Their primary responsibility is to oversee the infrastructure that supports the execution of various algorithms. This includes providing the necessary computational resources to ensure the smooth and efficient operation of the platform. In addition to maintaining the technical environment, the platform operator offers user support to assist users in navigating and utilising the platform effectively. They are also accountable for meeting the SLAs established for the platform, ensuring that performance and reliability standards are consistently met. While the operator may be a partner outside of the APEx consortium, their role is integral to the success of APEx, focusing on both operational management and user satisfaction.

3 ALGORITHM SERVICE DEVELOPMENT OPTIONS

APEx currently offers two main options for projects to develop their on-demand services:

- OpenEO-based services, implemented in line with openEO's User Defined Processes (UDP) [1]
- OGC API - Processes-based services, implemented in line with the OGC Application Package Best Practice [4]

These APEx-compliant technologies allow algorithms to be hosted on an APEx-compliant algorithm hosting platform and make them available for execution through web services. These technologies promote seamless reuse and integration of existing EO algorithms. Additionally, by leveraging web services and cloud-based approaches, these technologies simplify the execution of EO algorithms, shielding users from underlying complexities, such as data access, data processing optimisation, and other technical challenges.

APEx remains committed to future innovation and is open to integrating additional specifications, provided they align with FAIR principles and facilitate algorithm execution through web services.

The sections below provide a brief overview of each technology and outline various scenarios for their potential application.

3.1 openEO User Defined Processes

When an EO application can be expressed in terms of the standardised openEO processes (combined in a process graph), it can also be parametrised so that it effectively becomes a service that can be executed by an openEO backend. This is what we call a User Defined Process (UDP) [1].

3.1.1 When to use openEO User Defined Processes?

This option is a good choice when writing EO algorithms from scratch or when using Python libraries that support *numpy* or *XArray* data structures, which includes many machine learning frameworks. Therefore, it is also fairly common for projects to port their existing pure-Python algorithm into an openEO process graph. While this porting involves learning to understand the concepts of datacube-based processing and the openEO processes, it does provide some benefits.

The standardised, datacube-based processing approach is designed to significantly reduce the responsibilities of the EO data scientist writing the algorithm. The openEO backend takes care of data access, solving performance and stability issues while also dealing with the multiple heterogeneous formats that are used in earth observation. Backends will automatically parallelise the processing work using state-of-the-art technologies, as provided by Apache Spark or Pangeo.

Many common maintenance operations are performed by the openEO backend rather than by the EO service provider. This includes integrating performance improvements, new versions of software libraries with bug fixes, or adjusting to changes in the EO datasets.

Various complex preprocessing steps may be offered by the backend. Examples include backscatter computation for Sentinel-1 or cloud masking and compositing of optical data. Use cases like machine learning training data extraction are also supported.

When EO algorithms are expressed as process graphs, they also become easier to share and analyse by peers if the project chooses to make them public. Transparent algorithms can greatly help to increase confidence in the results or even to receive suggestions for improvements.

3.1.2 Examples

A set of platform-agnostic example cases that showcase how openEO supports various cases can be found at https://github.com/Open-EO/openeo-community-examples. APEx recommends browsing the notebooks available there to get a sense of what openEO can do or even to find a starting point for the project's own algorithm.

In addition to that, we list testimonials of ESA projects that successfully used openEO in the past.

3.1.2.1 ESA WorldCereal

The ESA WorldCereal project effectively built a system on top of openEO that enables the training of custom machine learning models for crop type detection anywhere in the world. The created models can then be used within openEO to generate maps at a large scale. The system is very easy to use because all the complex computation and data access is performed in the cloud.

The WorldCereal workflow for crop type map production contains these steps:

1. Generation on monthly Sentinel-2 median composites for 1 year of input data.
2. Generation of monthly Sentinel-1 backscatter composites from raw GRD input for 1 year of input data filtered by orbit direction.
3. Loading DEM, slope, and AGERA5 datasets.
4. Computation of features based on all the above inputs using a foundation model.
5. Inference using a (user-defined) CatBoost model.
6. Output to FAIR GeoTIFFs with STAC metadata.

3.1.2.2 ESA ANIN

The ESA ANIN project computed a combined drought index for South Africa using openEO. It successfully combined data from a variety of sources and applied rule-based techniques to compute the result.

3.1.2.3 ESA WorldWater

The ESA WorldWater project converted their decision algorithm based on Sentinel-1 and Sentinel-2 into openEO. This allowed them to run their water detection algorithm anywhere in the world using a cloud-based system.

3.1.2.4 ESA Luisa

Luisa used openEO to compute biophysical parameters such as LAI and fAPAR from Sentinel-2 data over various test areas in Africa.

3.2 OGC Application Package and OGC API – Processes

The OGC API processes specification [4] allows you to expose any type of (processing) web service. As a service designer, you get full freedom and responsibility in the definition of your service. This large degree of freedom also implies that the definition of your service will affect its interoperability with other services and tools.

APEx recommends that these services are built based on the OGC Application Package best practice [2]. This allows service providers to easily host their applications on a compatible hosting platform, taking away key IT challenges.

3.2.1 When to use OGC Application packages?

An OGC application package is a good choice when you have an existing piece of software that you want to make available as a service and do not aim to make substantial changes. In effect, it is a packaged version of your software, using (Docker) containers, which is a well-known technology for most IT professionals. A large amount of existing EO applications is known to already use containers. The container is invoked by a workflow written in the 'common workflow language' (CWL) [3], which is a thin wrapper around your software.

If concepts like containers are already familiar to the project, and they prefer to have full control and responsibility over their software, including all details concerning how they read raw EO products, then this might be the right choice. Of course, the APEx Algorithm Services exist to help you with the integration of your service into the APEx ecosystem.

To offer your application as an on-demand service, the project will need to define clear constraints on possible input parameters and make sure that their software is well-tested to be able to handle variations in input parameters. When changes occur, for instance, to raw EO data sources, the project will need to update its software accordingly, or it might no longer be compatible with the provided inputs. Likewise, if improvements are made to dependencies of the project's software, they will have the choice of updating its software or sticking to the old versions.

4 ALGORITHM DEVELOPER AND PROVIDER GUIDELINES

Table 1 outlines the interoperability prerequisites required for algorithm providers, such as EO projects, to host their workflows and algorithms within APEx. By satisfying these requirements, APEx guarantees the successful integration of workflows and algorithms and ensures reusability within the broader EO community. It is important to highlight that the majority of these requirements apply to EO projects that build an on-demand service to be exposed via an HTTP-based API.

In terms of creating on-demand services, APEx currently supports two main interface standards: openEO [5] or OGC API Processes [4], as described in section 3. This selection should support almost any possible on-demand service.

Table 2 provides an overview of additional recommendations that will simplify the integration of your algorithm into APEx. While these are not mandatory requirements, they are primarily suggestions aimed at enhancing the quality of your service for long-term reuse. Adhering to these recommendations will further streamline the integration of the service and improve its reusability, stability, and maintainability over time.

Finally, note that APEx also provides support to projects that need to fulfil these requirements. This support includes implementing the guidelines in this document, offering a framework to run automated tests, and providing packages to help improve the performance of algorithms. These are referred to as APEx Algorithm Services.

In general, the aim is to simplify the process of building high-quality on-demand services rather than to add complexity.

Table 1. Interoperability requirements for algorithm providers

ID	Requirement	Description
ALGO-REQ-01	The algorithms shall be provided according to one of these options: • Process graphs encapsulated in an openEO 'user defined process' (UDP) [1], as a JSON format, defined using the openEO community standard and using the openEO API [6]. • Applications written in a variety of coding languages (e.g., Python, R, Java, C++, C#, shell scripts) packaged according to the OGC Application Package Best Practice [2] as a CWL file [3].	This ensures that the algorithm can be hosted on one of the APEx-compliant algorithm hosting platforms. The APEx documentation will provide clear guidance and samples demonstrating these two options.
ALGO-REQ-02	Non-code dependencies such as custom datasets or machine learning models shall either be packaged with the software or be clearly listed as external dependency.	This approach prevents issues related to missing dependencies and ensuring users can easily set up the environment for proper execution. It also promotes transparency and simplifies the deployment process.
ALGO-REQ-03	Algorithms shall expose a list of well-documented parameters, with examples showing valid combinations of parameters.	Good documentation improves the usability of the algorithms by providing users with clear guidance on how to configure them correctly. Well-documented parameters and examples reduce the risk of incorrect usage. Depending on the used standard, this documentation should be added to the openEO UDP, CWL or OGC API Process.

ALGO-REQ-04	Executables shall offer at least one choice of a non-interactive command line interface, or an API for integration into a larger codebase.	A non-interactive command line interface or API enables seamless automation and integration with other components and services.
ALGO REQ-05	Algorithms shall be deployed and tested on at least one APEx-compliant algorithm hosting platform.	Testing and deployment on an APEx-compliant platform guarantees that the algorithm performs correctly within the intended environment and allows the algorithm to be executed as an on-demand service. APEx provides tooling to execute and verify results on compliant platforms through automated testing.

Table 2. Interoperability recommendations for algorithm providers

ID	Requirement	Description
ALGO-REC-01	Algorithms services should include documentation that addresses the scientific and technical limitations of the algorithm.	For instance, the ability of the algorithm to generalize across space and time, input data requirements, error margins on predicted physical quantities.
ALGO-REC-02	The algorithm provider should demonstrate the code quality via static code analysis tools.	For Python code, tools such as *pylint* can be used for static code analysis.
ALGO-REC-03	Validated outputs for a given set of input parameters should be made available by the algorithm provider, preferably on a small area that still allows for relevant testing.	This allows to validate the correct functioning of the algorithm as changes are made.

ALGO-REC-04	Automated tests should be provided that compare the current output of the software against a persisted sample, for a representative area of interest.	These tests enable APEx to automate the periodic validation of algorithms, ensuring they remain functionally available even after the project has finished.
ALGO-REC-05	A versioning scheme should be defined, preferably following a standardized approach such as https://semver.org. .	The versioning scheme provides a clear framework for communicating algorithm updates to users. By adopting standard practices, projects can highlight breaking changes, ensuring that users have accurate expectations and can adapt accordingly. The APEx documentation provides more examples and guidance on concrete cases.
ALGO-REC-06	The procedure for releasing new versions should be clearly documented.	Clear documentation ensures that updates and new versions of algorithms are consistently and correctly released, reducing errors and providing transparency for users who rely on the latest features and fixes. It also helps maintain version control, which is crucial for reproducibility and compliance. A common way to provide this documentation is to add it in the 'README' of the algorithm git repository.
ALGO-REC-07	For open source software developed within the project, a changelog should be maintained by the project.	This outlines significant changes between versions, providing important information for users of your algorithm and the APEx consortium. These explanations help clarify any differences in outcomes or performance that could impact automated testing.
ALGO-REC-08	Algorithms should clearly list software library dependencies, separated into testing, development, and minimal set of runtime	By clearly listing and categorizing dependencies, users can quickly set up the necessary environment, avoid

	dependencies. Supported versions or version ranges shall be indicated.	conflicts, and ensure the algorithm functions as intended across different scenarios.
ALGO-REC-09	Runtime dependencies should be minimized as much as possible.	For instance, libraries required for training a model should not be included in a version for inference.
ALGO-REC-10	Code should be written in a cross-platform manner, supporting at least Linux.	The support for Linux is considered crucial to enable the deployment of the code on a cloud-based environment.

4.1 Best Practices

The following sections provide best practice guidelines for developing APEx-compliant algorithms. While these guidelines are not mandatory, adhering to them will enhance the integration process and improve the overall experience of using the algorithm.

4.1.1 Parameter naming & typing

APEx proposes to standardise openEO UDP [1] and CWL [3] parameter names and types that are exposed to the user. This is best illustrated by an example: parameters such as *bounding_box*, *bbox*, *aoi*, and *spatial_extent* likely refer to the same concept. However, without common conventions, algorithms might randomly select one of these variants, complicating the usability of the eventual algorithm library.

At the time of writing, the actual conventions have not yet been defined. This becomes relevant when the first algorithms reach a state where they can be published with a fixed interface.
This best practice mostly targets new developments that do not have an existing user base or API.

4.2 Licensing Requirements

For algorithms to be hosted and curated, APEx requires the ability to execute the algorithms as on-demand services on an APEx-compliant algorithm hosting platform. This is straightforward for fully open-source algorithms, such as those licensed under the Apache 2.0 license. However, for algorithms with more restrictive licenses or those dependent on artefacts like trained machine learning models, the algorithm provider must be able to license APEx to execute the service without incurring additional costs beyond the normal resource usage. Without such a license, the automated benchmarking and testing provided by APEx may need to be disabled for the service in question.

5 DATA PROVIDER GUIDELINES

5.1 Requirements

Table 3 outlines the interoperability guidelines for EO projects and data providers who wish to deliver their datasets to APEx for integration within the ESA Project Results Repository (PRR) for long-term preservation and their utilisation within the APEx Project Environments. By fulfilling these requirements, APEx ensures seamless integration, discoverability, and usability of the datasets across the ESA EO ecosystem, facilitating broader access and reusability within the EO community.

Most of these requirements focus on standardising dataset metadata, formats, and access methods to ensure compatibility with existing tools and support their efficient exploitation. In particular, datasets should adhere to well-established EO data standards and provide consistent, machine-readable metadata descriptions.

APEx supports integration primarily through recognised standards such as STAC (SpatioTemporal Asset Catalogue) and cloud-native data formats. This ensures that almost any EO dataset can be made available as a ready-to-use resource in the ESA PRR and used through the APEx tooling.

Overall, the objective is to streamline and simplify the delivery of high-quality, interoperable EO datasets to APEx, fostering wider adoption and enabling advanced use cases in downstream applications.

Table 3. Interoperability requirements for data providers

ID	Requirement	Description
DATA-REQ-01	EO project results with respect to raster data, shall be delivered as cloud-native datasets.	Where possible, cloud optimized GeoTIFF [7] is preferred. For more complex datasets, CF-Compliant netCDF [8] is a good alternative. Use of the still evolving GeoZarr [9] format requires confirmation by APEx and may result in future incompatibility if the selected flavour is not standardized eventually. Additional recommendations for the usage of file formats within the APEx services is available on the APEx documentation.
DATA-REQ-02	EO project results with respect to vector data, shall be delivered as cloud-native datasets.	Small datasets can use GeoJSON [10], FlatGeobuf [11] or GeoParquet [12] are recommended for larger datasets.
DATA-REQ-03	EO project results should be accompanied with metadata in a STAC [13] format, including applicable STAC extensions.	The specific STAC profiles will align with the recommendations provided by the ESA Project Results Repository (PRR). More details regarding which profiles to apply will be added as the project progresses.

5.2 File Format Recommendations

Within the APEx interoperability and compliance guidelines, DATA-REQ-01 specifies the file formats that projects should use when publishing static data. The information on this page complements that requirement with more detailed recommendations on the use of these formats. Projects are encouraged to consult with the APEx team if they have a use case that does not fit within these guidelines.

5.2.1 Cloud Optimised GeoTIFF

Cloud Optimised GeoTIFF (COG) is the most widely supported format for geospatial raster data, and also one of the most efficient in terms of access costs. It is recommended as the default option to consider when publishing static data products. When combined with STAC metadata, COG can produce self-describing, FAIR-compliant datasets that are easily consumable by various services.

5.2.1.1 Generating Cloud Optimised GeoTIFF

If you are not familiar with COG or GeoTIFF generation, it is recommended to format your GeoTIFF files using a recent version of GDAL to ensure that they are compliant:

gdal_translate world.tif world_webmerc_cog.tif -of COG

5.2.1.2 Organising spatiotemporal multi-band datasets

Many datasets have multiple bands (or ‘variables’), or have a date associated with them. The general recommendation is to store a single band per GeoTIFF file and to create STAC items with one asset per band. This layout is commonly used by many other datasets and avoids the complexities of multi-band GeoTIFF files, which can be challenging to use.

An exception to this are the ‘RGB’ style products, where three bands are used to represent a single image. In this case, creating a Cloud Optimised GeoTIFF with three bands is an option.

For associating time information, create one GeoTIFF per timestamp, and one STAC item per timestamp. The GeoTIFF format has not built-in support for conveying time information, but STAC metadata is supporting this very well.

5.2.2 Visualisation in APEx Geospatial Explorer

To optimise visualisation in the APEx Geospatial Explorer, additional guidelines have been established, as outlined in section 7.2. Adhering to these guidelines will ensure that the data is effectively optimised for visualisation on a map.

5.2.3 (Geo-)Zarr

Zarr is a format that is gaining traction in the geospatial community, although it is not yet as widely supported as Cloud Optimised GeoTIFF. Its main advantage lies in its ability to store complex multi-dimensional datasets that go beyond a simple 4D (x, y, time, bands) structure. Just like COG, Zarr allows for efficient cloud access.

At the time of writing, there are, however these important caveats:

- GeoZarr aims to define how to store geospatial data in Zarr format, but this standard is not fully developed and lacks widespread tooling support.
- Overview pyramids for fast online rendering are not yet supported.
- By design, Zarr allows for many degrees of freedom, which requires the data producer to have a good understanding of the associated trade-offs.
- By design, Zarr stores data as separate files in a directory structure, optimising cloud access but making direct downloads less convenient.

5.2.4 NetCDF

NetCDF is a self-describing format with some properties similar to Zarr, but less optimised for cloud access. It can be useful for exchanging data cubes as single files through traditional methods. However, it is less recommended for convenient sharing of large datasets, for which either COG or Zarr provide better options.

6 ALGORITHM HOSTING PLATFORMS GUIDELINES

6.1 Requirements

APEx-compliant application algorithms are executed as services on a compatible algorithm hosting platform. Currently, APEx supports platforms based on either openEO [5] or OGC Application Package [2] technologies. To be considered an APEx-compliant algorithm hosting platform, the platform must meet the requirements outlined in this document section.

The aim of these requirements is:

- To ensure that services developed in EO projects continue to work after the project has ended.
- To incentivise EO platforms to become more interoperable and become part of an interoperable federation.
- To align algorithm hosting platforms that aim to offer EO project services to make them more comparable. The alignment targets the API level and the (high-level) pricing model, giving platform providers full freedom to select technologies and architectures that suit their needs.
- To allow APEx to perform automated checks on the developed services, guaranteeing that they work and produce the expected result at the expected cost.

Table 4 provides an overview of the requirements for operators of algorithm hosting platforms to ensure their compatibility with the APEx guidelines.

Table 4. Interoperability requirements for algorithm hosting platforms

ID	Requirement	Description
HOST-REQ-01	The algorithm hosting platform shall support the generation of cloud-native datasets, such as Cloud Optimized GeoTIFF [7], CF-Compliant netCDF [8].	Support of GeoZarr [9] will become a requirement as soon as a clear standard is available, check with APEx for guidance on current variants.
HOST-REQ-02	The algorithm hosting platform shall support the generation of metadata in a STAC [13] format for both raster and vector data, including applicable STAC extensions.	This requirement ensures that the outputs are compatible with modern geospatial data standards, enhancing interoperability with other platforms and services.
HOST-REQ-03	The algorithm hosting platform shall support at least one of these API's: • OpenEO API version 1.2 [6] – At least API profile L1B-minimal-batch-jobs [14] – At least processes profiles L1-minimal [15] • OGC API Processes [4], supporting the deployment of applications defined by the OGC Best Practice for Earth Observation Application Package [2].	The support for the proposed technologies ensures that APEx-compliant algorithms can be hosted on the algorithm hosting platform and made available as on-demand services.
HOST-REQ-04	The algorithm hosting platform shall be registered in the ESA Network of Resources.	To curate services beyond the project lifetime, the APEx consortium is likely to require an active subscription on the platform. For paid subscriptions, this can only be requested via NoR sponsoring. This may not be necessary if the platform is willing to grant non-paid subscriptions.

HOST-REQ-05	The algorithm hosting platform shall provide an SLA that guarantees support beyond the project lifetime.	This ensures the long-term sustainability and reliability of the algorithm hosting platform, providing assurance to users that they can rely on continued support even after the project's completion.
HOST-REQ-06	The operator of the algorithm hosting platform shall announce major changes to the SLA, including decommissioning of the platform, to APEx and the NoR, preferrably with a lead time of 1 year.	Such communication is important to ensure that stakeholders, including APEx and the NoR, are given adequate notice of major changes that could impact the availability or functionality of the algorithm hosting platform. This approach allows for proper planning, adjustment, and mitigation of potential disruptions, ensuring continuity of services for users.
HOST-REQ-07	The algorithm hosting platform shall support standardized methods for machine-to-machine authentication.	For instance, the OIDC client credentials workflow allows APEx to securely authenticate.
HOST-REQ-08	The operator of the algorithm hosting platform shall support the APEx consortium in obtaining a single account either freely, or via ESA Network of Resources, that allows to test all published services.	For convenient service testing and minimal administrative overhead, a single account and NoR request should give access to multiple services.
HOST-REQ-09	The algorithm hosting platform shall expose process metadata publicly without requiring authentication, unless the nature of the algorithm requires its description to be hidden.	APEx tools request service metadata for informative purposes, also from browser based applications that do not have platform tokens available.

6.2 openEO API Specific Requirements

The algorithm hosting platform implementing openEO API [6] shall support applications defined as openEO User Defined Processes (UDP) [1]. In the context of APEx, these will be executed as openEO batch jobs. The minimal requirements for an openEO algorithm hosting platform to support this feature are described in the profile called L1B-minimal-batch-jobs. The openEO API, provided by the algorithm hosting platform, shall support at least the L1-minimal processes profile.

6.2.1 Open Source UDPs

For open-source UDPs, APEx will use the openEO remote UDP extension, which is currently under review by the openEO community. This extension enables APEx to centrally store and manage the UDP definitions, using the algorithm hosting platform only for the actual processing. This is to ensure consistent management of UDP definitions, safeguarding them from loss if the algorithm hosting platform is decommissioned.

We assume that this is the default case because EO projects have a strong preference for open-source software.

6.2.2 Private UDPs

The specification of openEO allows platforms to expose custom processes that function like openEO UDPs but do not expose the underlying process graph. This situation may arise for two main reasons:

- The process graph exists, but the project is not required to share it publicly. In this case, we assume that this is supported by ESA, and it is considered good practice to use openEO.
- The process graph does not exist, and the process triggers an arbitrary processing system. In this case, the OGC Application Package approach might be a better alternative.

6.3 OGC Application Package Specific Requirements

EO Application Packages, defined according to the OGC Best Practice [2], use the Common Workflow Language (CWL) [3] for process descriptions and are often containerised (e.g., using Docker) to ensure portability and consistency. To host and execute these packages effectively, platforms must meet a set of key requirements, particularly around data handling and interoperability, such as:

- Support for Application Deployment: Platforms must be capable of interpreting CWL-based workflows and deploying the associated containerised environments.
- Data Staging Using STAC: Efficient stage-in (input data retrieval) and stage-out (output data publication) are essential for EO workflows. Platforms should integrate with STAC to:
    - Discover input datasets through STAC-compliant APIs or catalogues.

    - Facilitate the publication of results in a STAC-compliant format for downstream use.
    - Ensure metadata-rich, standardised access to EO data, streamlining workflows.
- Resource and Process Management: Platforms should provide scalable computational resources to execute applications efficiently. Resource management might include job scheduling, monitoring, and optimising containerised environments.
- Standardised API Support: Platforms should implement the OGC API - Processes [2] to allow consistent interaction with workflows, enabling users to manage jobs and execute processes

Guidelines and recommendations to set up a cloud environment that is compliant with this approach can be obtained directly from the EOEPCA project (https://eoepca.org). The project includes documentation, software and deployment configurations (e.g. helm charts) to cover the needed platform requirements.

6.4 Service Level Commitments

The service level commitments for algorithm providers and algorithm hosting operators will be further documented by the work done in WP4/5 and WP9/10.

6.5 Supported Platforms

This section provides an overview of APEx-compliant algorithm hosting environments based on the APEx interoperability and compliance guidelines. The list includes platforms available for each APEx-compliant technology, along with recommendations for environments that support the development process of APEx-compliant services.

Throughout the project, APEx aims to expand this list of compliant platforms, offering users the flexibility to choose their preferred platform.

6.5.1 openEO User Defined Processes

An openEO UDP can be developed in any openEO-based environment using existing openEO tools. APEx can support development activities by providing projects access to its User Workspace and IDEIDE services. For testing the UDP, we recommend utilising an APEx-compliant hosting platform. This ensures platform compatibility and facilitates the deployment of your UDP as an on-demand service. Currently, APEx has identified the following platforms as compliant with the APEx guidelines:

- openEO platform
- CDSE openEO federation

6.5.2 OGC API Processes

To develop and test an OGC Application Package, you can use one of the following environments, which include the necessary tools and basic infrastructure to support the development process:

- ELLIP
- APEx User Workspace and IDE services

To offer your algorithm as an OGC API Process, APEx has identified the following platforms as compliant with the APEx guidelines:

- Geohazards TEP

7 GEOSPATIAL EXPLORER

This section outlines the requirements for the interoperability of the APEx Geospatial Explorer services. These requirements must be met to ensure the correct configuration and operation of a dashboard and its instantiation.

A technical challenge of the Geospatial Explorer service being provided by APEx is that it is to be instantiated on demand, with functional requirements potentially varying amongst each service. A proposed solution to this challenge is the use of a well-defined configuration schema, provided in the form of JSON, that outlines the interactive features and data sources to be used.

This approach will allow APEx to define and update the schema required in the interoperability guidelines, which will then enable requesters of the service to configure the Geospatial Explorer on an individual project level with minimal external intervention.

The schema will be versioned as it will change throughout the APEx project as the functional capabilities of the Geospatial Explorer mature. This does allow for improvements and extra features to be easily added to the application, and best practices shall be followed to avoid any breaking changes between versions.

7.1 Requirements

Table 5 outlines the requirements for configuring an instance of the Geospatial Explorer application.

Table 5. Requirements for configuring Geospatial Explorer

ID	Requirement	Description
EXPLORER-REQ-01	Configuration of the Geospatial Explorer shall adhere to the provided JSON schema, as described in section 7.2.	The application is configured through a hosted JSON object which is fetched when the client-side application loads. For the application to correctly render, a valid configuration JSON that follows the outlined schema must be provided pre-instantiation. During early development the schema will be subject to change and provided with limited validation.
EXPLORER-REQ-02	Raster sources defined within the config shall be provided as either: • A well formed URL to a hosted Cloud Optimised GeoTiff (COG) [7] • A WMS [16] or WMTS [17] endpoint following the relevant OGC specifications. • A template URL following the XYZ[1] format serving PNG or JPEG tiles.	The use of these formats ensures that raster data can be visualized using widely recognized and well-established standards.

[1] The XYZ approach refers to a *de facto* API standard for URL structuring (Z = zoom level, X and Y = grid references). While there is no dedicated specification for XYZ, it is widely used (e.g., OpenStreetMap - https://wiki.openstreetmap.org/wiki/Raster_tile_providers).

EXPLORER-REQ-03	Vector sources defined within the config shall be provided as either: • A well formed URL to a valid GeoJSON [10] resource • A public WFS [18] or OCG API Features [19] endpoint following the relevant OGC specifications • A well formed URL to a valid FlatGeobuf [11] resource	The use of these formats ensures that vector data can be visualized using widely recognized and well-established standards.
EXPLORER-REQ-04	All external sources shall be publicly available.	Currently the application does not support fetching sources from hosts that require authentication.
EXPLORER-REQ-05	All external sources shall be available utilising the HTTPS protocol.	It is best practice to utilise HTTPS where possible. Whilst not as of yet blocked specifically, non-secure resources may not function correctly.
EXPLORER-REQ-06	Web service endpoints for data must be configured to allow cross-origin requests from the domain hosting the Geospatial Explorer.	This includes setting the Access-Control-Allow-Origin header to explicitly permit the origin of the requesting front end or using a dynamic or wildcard origin setting if appropriate. Additionally, the web service must handle preflight (OPTIONS) requests correctly, including the Access-Control-Allow-Methods and Access-Control-Allow-Headers headers, to support the HTTP methods and headers used by the front end.
EXPLORER-REQ-07	All datasets shall be projected to CRS EPSG:3857 (Web Mercator); EPSG:4326 (WGS84) or EPSG:3035 (ETRS89-extended / LAEA Europe)	Other projections may work but will not be explicitly supported in the initial versions of the application. Support for these projections will be considered and reviewed on a case-by-case basis and will be subject to support by the PROJ4JS library [21]

EXPLORER-REQ-08	Legends should be configured by providing: • An inline configuration using either the 'Swatch' or 'Gradient' outlined in the schema. • A URL to a browser supported image resource (PNG, JPEG, SVG). • A WMS [16] getLegendGraphic request (If supported by a WMS source).	This guideline ensures that the legend is fully compatible with the visualisation library used in the Geospatial Explorer.
EXPLORER-REQ-09	Statistical datasets should be configured by providing public URLs to either: GeoJSON [10] or FlatGeobuf [11]	The statistics feature is complex due to the architecture and requirements of the Geospatial Explorer. One or more files containing vector features that describe an area and the relevant statistics for the area can be added to the configuration of an explorer instance. These features require a specific structure of properties that can be found in the APEx GE Documentation

7.2 Format Specification & Guidelines

The following chapters outline guidelines and specifications for using specific file formats in the Geospatial Explorer. For more details, please refer to the APEx documentation portal: APEx File Format Documentation.

7.2.1 Cloud Optimized GeoTiff (COG)

When generating Cloud Optimised GeoTiffs, it is recommended to use the *GoogleMapsCompatible* tiling scheme-typically 256x256 pixel tiles aligned to a global grid-and to store the image in the Web Mercator projection (EPSG:3857). The *BitsPerSample* field must accurately reflect the data format. Overviews are essential for performance and should be generated using downsampling by factors of two until the image dimensions are the size of a tile or smaller. These overviews should also be tiled and placed after the main image data to conform with the COG specification.

7.2.2 Statistics (Vector Layers)

The statistics feature expects vector layers that are provided in a format that can be parsed to a feature collection following the GeoJSON [10] specification. Currently tested and supported formats are GeoJSON [10] and FlatGeobuf [11]. FlatGeobuf should be used where the statistical data is a large size as this allows for streaming of the relevant features without having to download the full dataset, increasing performance.

7.3 Configuration Schema

The service configuration will be based on a schema that provides administrators with the expected structure and contents of the configuration. Taking this approach enables:

- Automated and dynamic instantiation of the service with differing functionality.
- Configuration validation.
- Definition of a “contract” for easier documentation of features and their configuration.

For more detailed information on this configuration schema, refer to the APEx documentation portal: APEx Configuration Schema Documentation. Numerous example configurations can be found in the APEx Geospatial Explorer Configurations repository on GitHub.

8 FEDERATED BUSINESS MODEL

This section outlines the commercial aspects of algorithm hosting within APEx and will be further analysed and documented by T6.1 Business Plan Development and T11.1 NoR Onboarding.

8.1 Algorithm Hosting Platforms

Table 6 provides an overview of the requirements and guidelines for the algorithm hosting platforms to support commercial algorithm hosting through ESA's Network of Resources.

Table 6. Requirements for supporting commercial algorithm hosting for algorithm hosting platforms

ID	Requirement	Description
BM-REQ-01	The algorithm hosting platform shall allow a fixed price per area unit.	This requirement ensures transparent pricing for the user, for example expressed in platform credits per km², hectare or by number of products.
BM-REQ-02	The operator of the algorithm hosting platform should support registering the algorithm in the ESA network of resources, with a cost per km², hectare or by number of products.	This does not apply if the service can be invoked by the user 'for free', so costs are compensated by a different mechanism.

vision on technology
for a better world